Reshaping Text Data for Efficient Processing on Amazon EC2

Gabriela Turcu, Ian Foster, Svetlozar Nestorov

THE UNIVERSITY OF CHICAGO

Outline

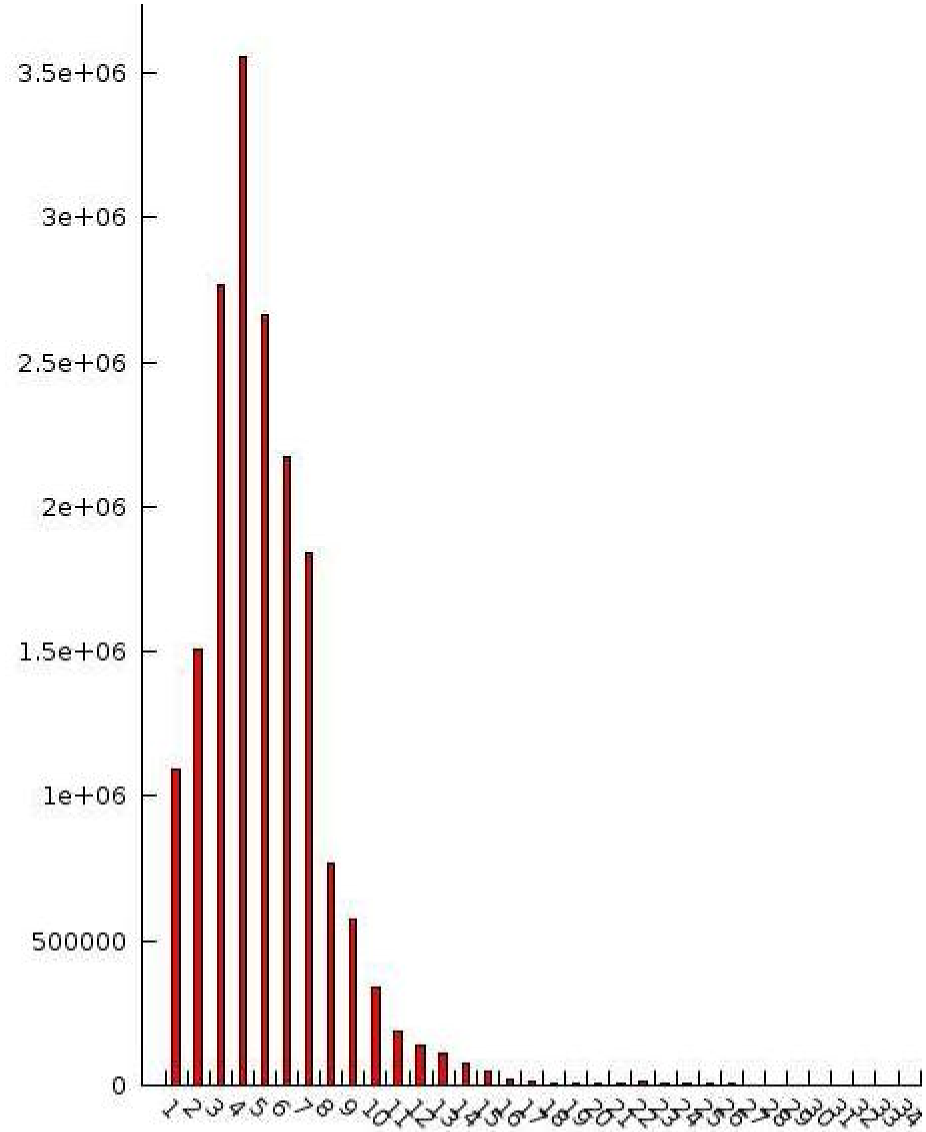
- Motivation
- Goals:
  - Determine empirically simple application performance model
  - Statically provision resources to meet user constraints
  - Reshape the input to avoid the small file problem
- Approach
  - Sample Applications – grep, part of speech tagging
- Summary

Motivation

- Analysis of large corpora
  - Online news collections
  - Text generated by social networks – tweets, status updates, comments, reviews
  - Scientific article abstracts, posters, slides

Text Datasets

- Heavy tail distribution
  - Majority of files of a few KB



Text processing in the cloud

- The analysis of large corpora demands increasing computational resources:
  - Cloud computing offers benefits:
    - On-demand provisioned environment
    - Pay-as-you-go pricing model
    - Customizable virtual machines that can be easily configured to incorporate legacy software
  - …and drawbacks:
    - Infrastructure controlled by provider
    - Environment volatility

Setting

- We have a large text workload, comprising of small files whose size distribution we know
- We do not have a model for the application performance in the cloud

- Can we construct empirically an application performance model to help provision resources within user constraints?
- Can we reshape the input data for improved performance? If so, what is the best organization?

Amazon EC2

- On-demand resizable computing capacity with a pay-as-you-go pricing scheme
  - Instances (small,medium,large) with different CPU, memory and I/O performance
  - AMI (Amazon machine images) with different configuration (32/64-bit architecture, Fedora/Windows/Ubuntu)

Amazon EC2 - storage

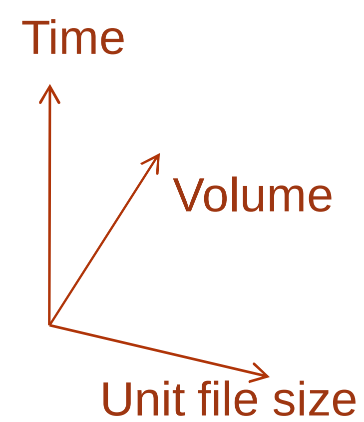
- Ephemeral
  - Instance store - 160GB for a small instance
- Persistent
  - Elastic Block Store (EBS) volumes
    - 1GB to 1TB in size
    - Exposed as raw block devices that can be 'attached' to instances
    - Cannot be shared between instances
    - Pay per GB/month and also per 1M I/O requests
  - Amazon Simple Storage Service (S3)
    - Unlimited number of objects up to 5GB each
    - Multiple instances can access this storage in parallel with low latency (though higher and more variable than EBS)

Approach

- Request instance (small, FC8) and measure its read/write performance
- Send probes of increasing volume to profile application
  - Send probe of 1 file of volume V_0 : P_ V_0 _original
  - Select larger volume V_1 as a multiple of V_0
    - Create P_V_1_original
    - Select base unit sizes (s_0,…,s_n)
    - Create P_V_1_s_0, …,P_V_1_s_n

    using first fit binpacking
      - Repeat binpacking for each probe
      - Create P_V_1_s_0 and then merge

      to obtain remaining probes – sensitive to quality of s_0 probe



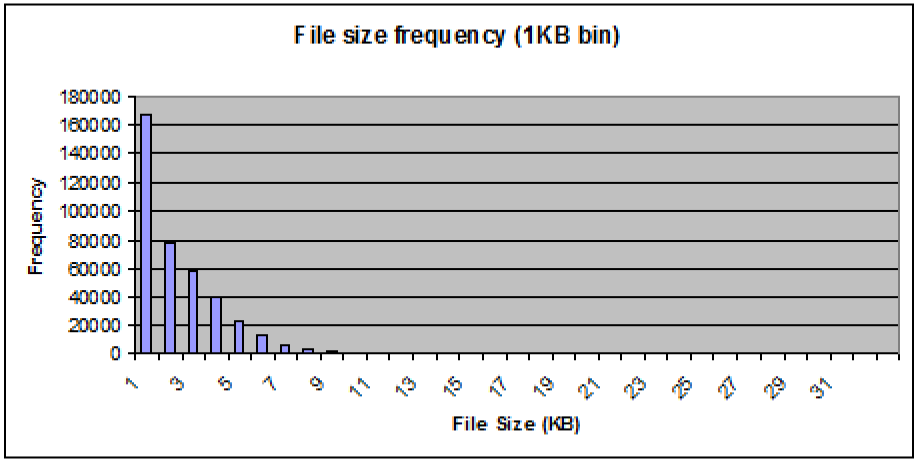
Approach

- If possible, select a unit file size that minimizes the execution time
- Reshape the data set according to match the file size as closely as possible
  - Splitting of a file not considered
- Derive a performance model as “execution time=f(vol)” performing linear regression on the measurements corresponding to the selected file size
  - Linear $y=ax$
  - Power law $y=ax^b$
  - Exponential $y=ae^{bx}$

Part of Speech tagging

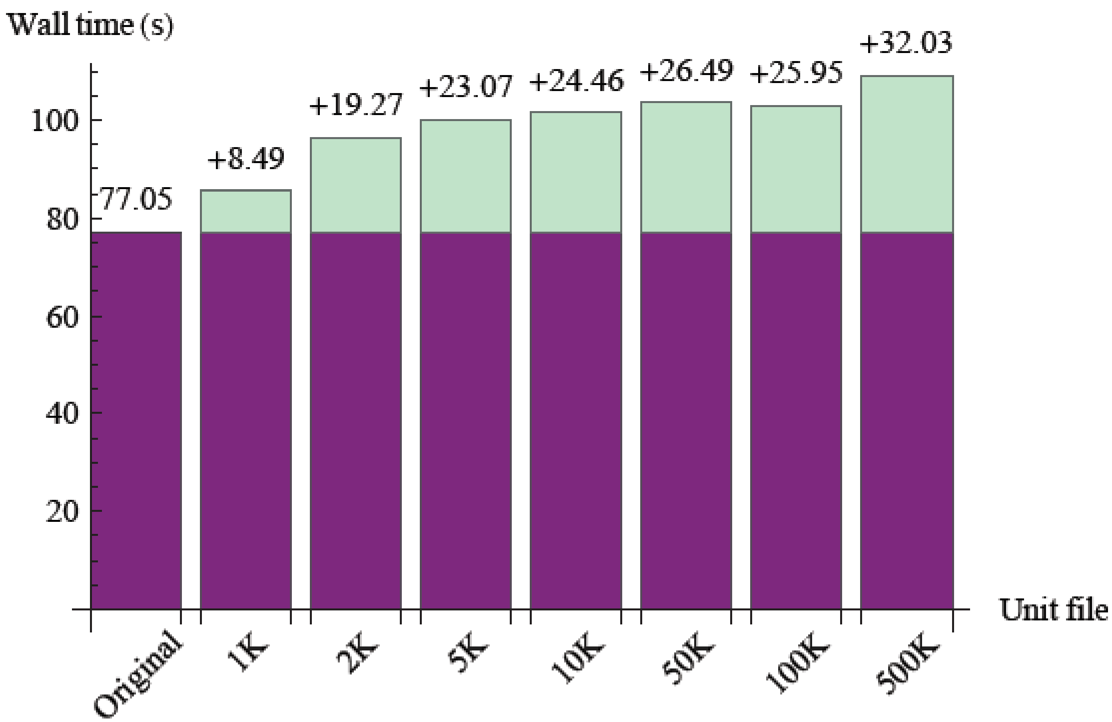
- Java implementation Stanford NLP POS tagger:

  Mary_NNP has_VBZ 3_CD little_JJ lambs_NNS ._.
- Process multiple files within same JVM
- Data set:
  - 1GB of text data
  - >40% of files are <1KB
- Small instance, instance storage



POS tagging

- V=1MB
- S_0=1KB, …, S_n=500k



» Original size performs best

Performance Model

- Linear fit

$$f(x) = 0.327 + 0.865 * 10^{-4} * x$$

- Solve for a deadline D=3600 (seconds) and obtain x_0 – the volume of data predicted to be processed within D
- For volume V, provision to meet the deadline:

$$i_0 = \lceil \frac{V}{\lfloor x_0 \rfloor} \rceil$$

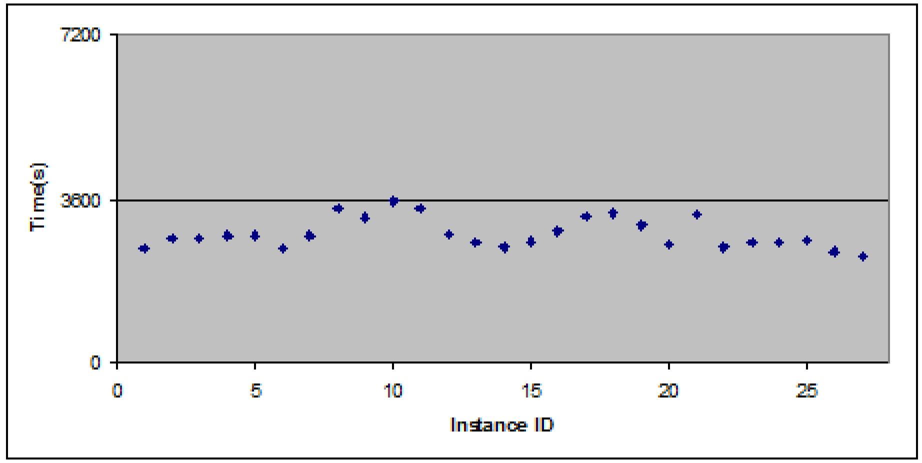
Static provisioning

- Bin packing for i_0=27 instances:
  - Sorted by file size
    - Better fit, but fewer large files in the initial bins – performance was bad for larger files
  - Taken as presented
    - More likely to get a balance between # of files and size

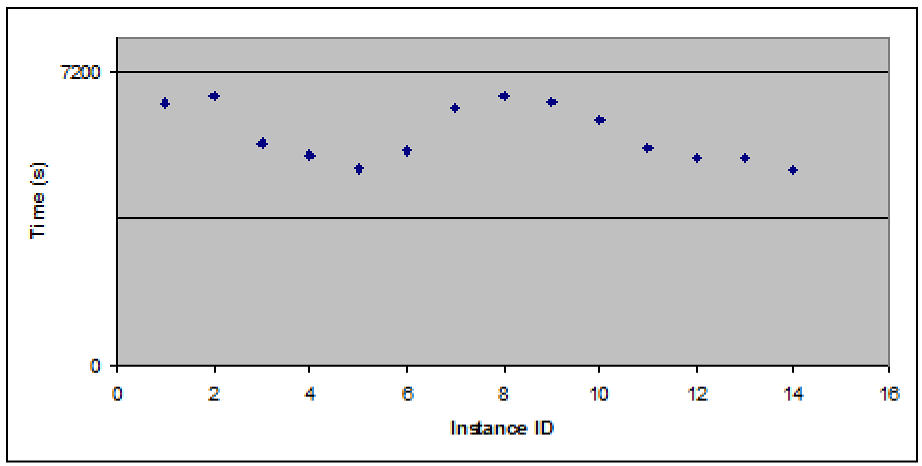
      » Other options can be explored

Initial results

- D=3600



- D=7200



» We could use fewer instances.

Random sampling

- Take random samples from the data and reevaluate performance model
- 3 samples of 5MB – profile each sample

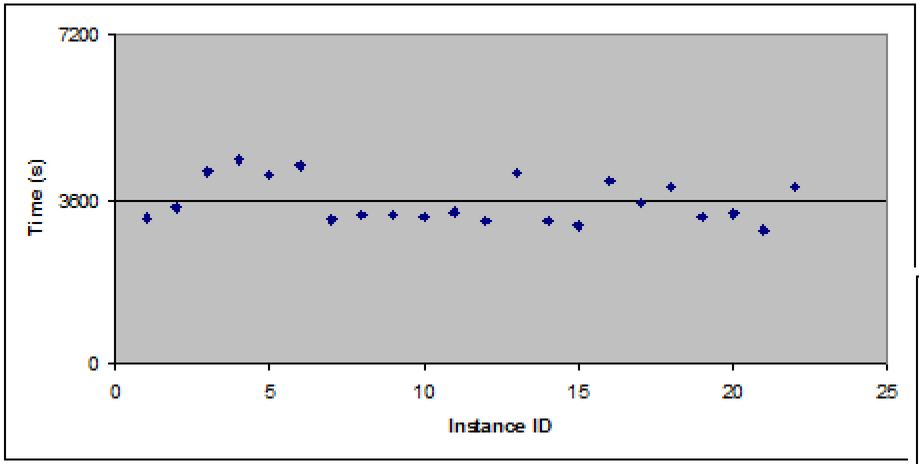
$$f(x) = 3.086 + 0.72 * 10^{-4} * x$$

  - The new slope is lower than the previous

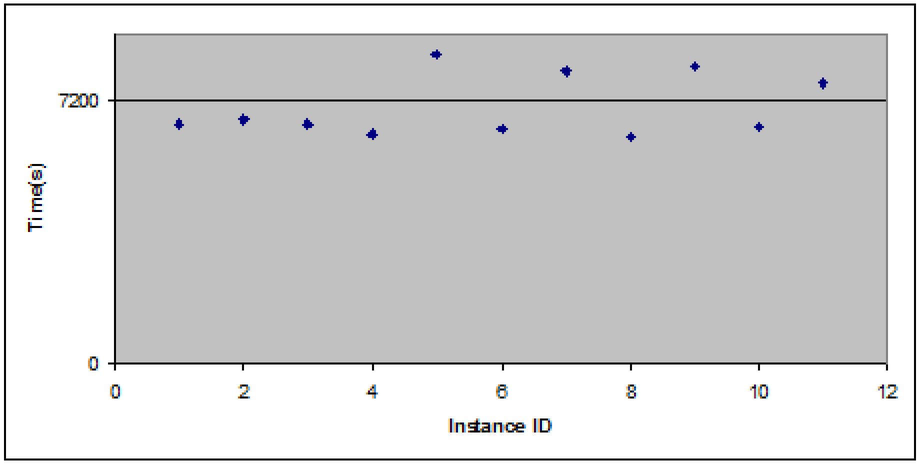
$$f(x) = 0.327 + 0.865 * 10^{-4} * x$$

POS tagging – random sampling

- D=3600



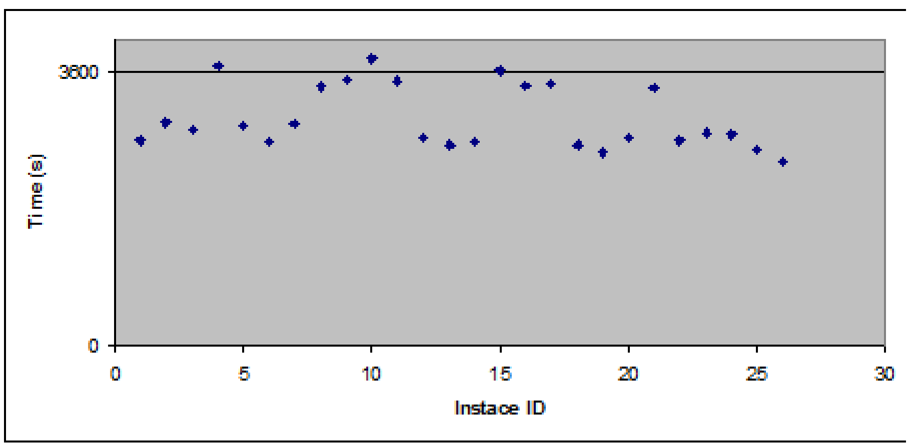
- D=7200

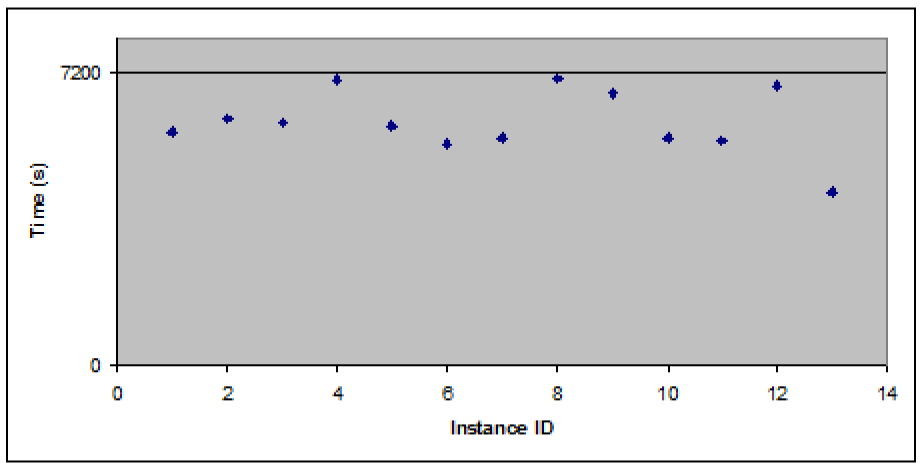


» Tighter fit, but we overshoot the deadline!

POS tagging

- We provisioned instances to exactly meet the deadline D (based on the model)
  - Residuals are can be considered normally distributed
  - Confidence interval analysis leads us to lower the deadline we provision for D=3600 -> 3124



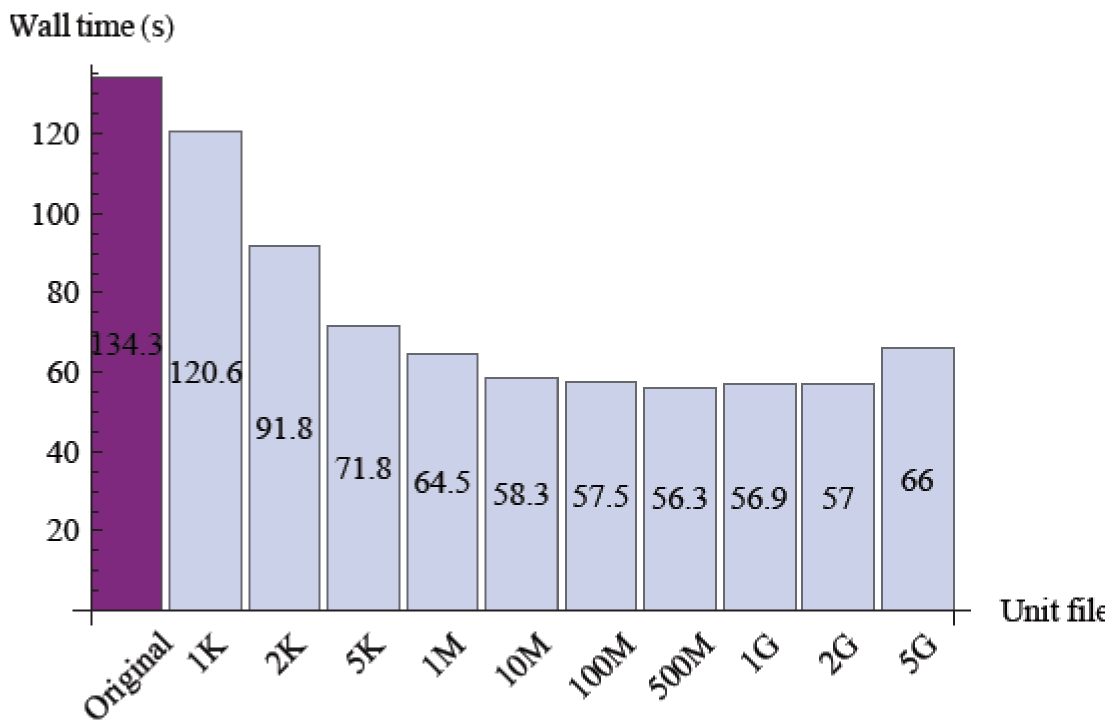


Grep

- GNU grep 2.5.1
- 100GB set of HTML files
- EBS storage
- CPU – I/O mix influenced by complexity of the search pattern
  - » Search for simple patterns – dictionary words
- Certain search modes and/or the likelihood of finding a match influences the amount of output generated
  - » Search for a nonsense word to traverse the entire input, but not generate output

Determining file unit size

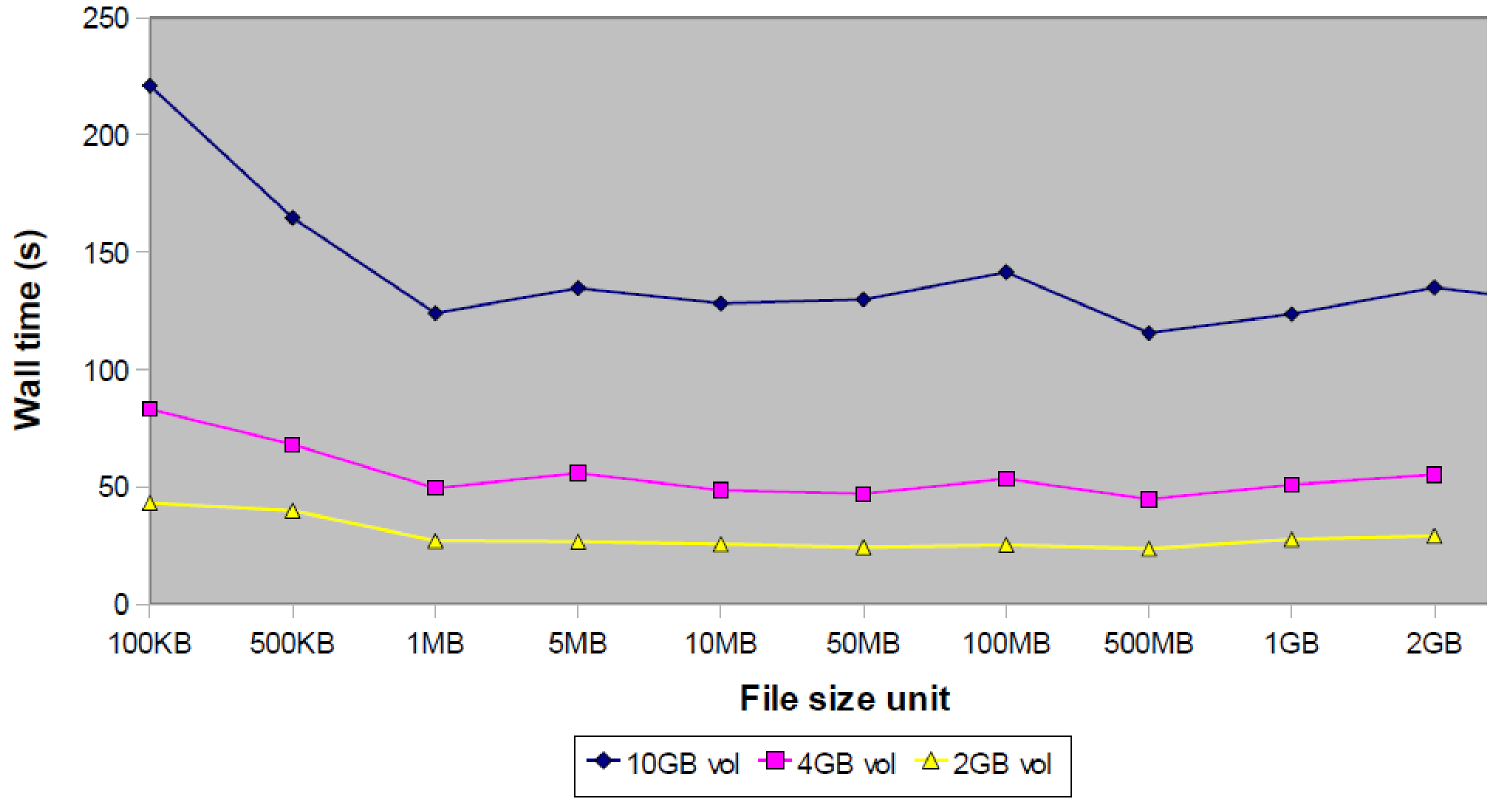
- V=5GB



» 1M-2G range performs well

EBS performance

- Plateau not smooth
- EBS performance consistently worse for some data sets

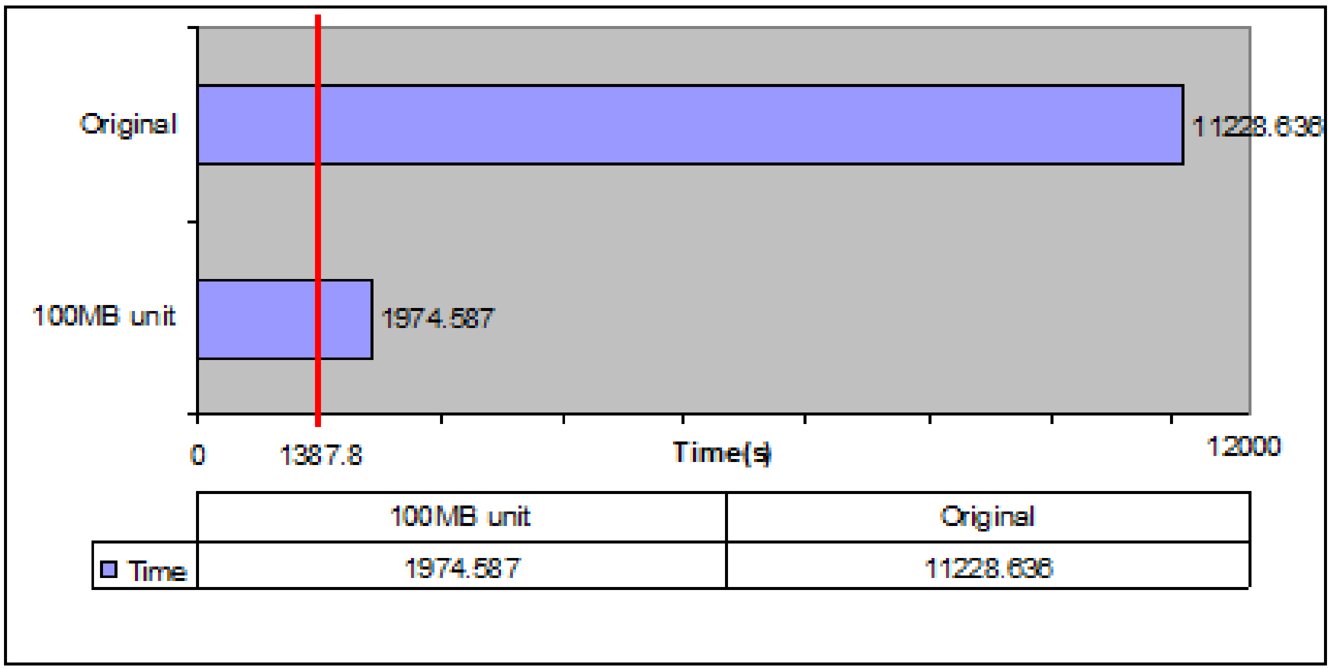


Provisioning

- If the fragment volume > predicted volume
  - Increase fragmentation level
- Othwerwise,
  - Attribute as much data to an instance as permitted by fragment volume multiples that fit into

Results

- Model: $f(x) = -0.97 + 1.32 * 10^{-8}x$
- D=3600



Summary

- Small scale experiments to learn application behavior on externally managed environment
- Determine if reshaping of input data set is beneficial
  - Grep – I/O intensive, reshaped to larger file sizes
  - POS tagging – memory intensive, reshaping not helpful
- Provision statically to meet user deadlines

Thank you!